AD-A094 786 WASHINGTON UNIV SEATTLE DEPT OF COMPUTER SCIENCE F/G 9/2
OPTIMAL PLACEMENT OF IDENTICAL RESOURCES IN A DISTRIBUTED NETWO--ETC(U)
JAN 81 M J FISCHER, N D GRIFFETH, L J GUIBAS N00014-80-C-0221
UNCLASSIFIED TR-81-01-03 NL

1 OF 1
AD A
094786

END
DATE
FILMED
3-81
DTIC

AD A094786

1

OPTIMAL PLACEMENT OF IDENTICAL RESOURCES IN A DISTRIBUTED NETWORK[†]

by

Michael J. Fischer
Nancy D. Griffeth
Leo J. Guibas
Nancy A. Lynch

Technical Report #81-01-03

[†]This research was supported in part by the National Science Foundation under grants MCS77-02474, MCS77-15628, MCS78-01698, MCS80-03337, U.S. Army Research Office Contract Number DAAG29-79-C-0155, and the Office of Naval Research Contract Numbers N00014-79-C-0873 and N00014-80-C-0221.

(15) N00014-80-C-0221

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(14)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR-81-01-03	2. GOVT ACCESSION NO. AD-A094786	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) OPTIMAL PLACEMENT OF IDENTICAL RESOURCES IN A DISTRIBUTED NETWORK		5. TYPE OF REPORT & PERIOD COVERED (7) Technical rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Michael J. Fischer, Nancy D. Griffeth, Leo J. Guibas [redacted] Nancy A. Lynch	(11)	8. CONTRACT OR GRANT NUMBER(s) ONR: N00014-79-C-0873 & N00014-80-C-0221; NSF MCS77-02474, MCS77-15628, MCS78-01678, MCS80-03337; ARO: DAAG29-79-C-0155.
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science, FR-35 University of Washington Seattle, WA 98195		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 049-456/30 Oct 79 (437)
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research, 800 N. Quincy, Arlington VA 22217, ATTN: Dr. R.B. Grafton / NSF, Washington, D.C. 20550 / U.S. Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709	(11)	12. REPORT DATE Jan ~~uary~~ 1981
		13. NUMBER OF PAGES 16
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research 801 N. Quincy Arlington, VA 22217 ATTN: Dr. R. B. Grafton	(12) 18	15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distributed unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer network, distributed system, efficient placement algorithm, optimality, resource allocation problem.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The problem is considered of locating a number of identical resources at nodes of a tree so as to minimize the total expected cost of servicing a set of random requests for the resources. The cost of servicing a request is the tree distance from the requesting node to the node at which the resource satisfying the request is located. An algorithm for finding an optimal placement of t resources is presented, which runs in time $O(t^2 \cdot |E|)$, where E is the edge set of the tree. In the special case of

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

315224 [illegible]

OPTIMAL PLACEMENT OF IDENTICAL RESOURCES IN A DISTRIBUTED NETWORK[†]

Michael J. Fischer
University of Washington

Leo J. Guibas
Xerox Palo Alto Research Center

Nancy D. Griffeth
Georgia Institute of Technology

Nancy A. Lynch
Georgia Institute of Technology

ABSTRACT

The problem is considered of locating a number of identical resources at nodes of a tree so as to minimize the total expected cost of servicing a set of random requests for the resources. The cost of servicing a request is the tree distance from the requesting node to the node at which the resource satisfying the request is located. An algorithm for finding an optimal placement of t resources is presented which runs in time $O(t^2 \cdot |E|)$, where E is the edge set of the tree. In the special case of a complete binary tree with requests uniformly distributed over the n leaves, another algorithm works in time $O(t \log_2 n)$. While optimal placements in general seem difficult to characterize, there is a very simple placement whose *total cost differs from optimality* by at most $|E|$. The expected cost of this placement on a complete binary tree is $O(n + \sqrt{tn})$.

1. INTRODUCTION

We consider the problem of locating some number t of identical resources at nodes of a distributed network in such a way as to minimize the expected "cost" of servicing a random set of t requests for those resources. Various different costs are likely to be important in different situations.

For example, suppose the resources are processors and requests are to establish a virtual connection with a processor which will be used for a very long period of time. In this case one might want to minimize the expected total of all the network distances (measured in some appropriate way) between requesting users and their assigned processors. Because of the long holding times, it is probably reasonable to expend considerable effort to find a good matching of requests to resources. The total of the network distances provides a measure of the expected communication traffic introduced into the network by the computations.

For another example, suppose the resources are tickets to sporting events (or airline seats). A relevant cost is the expected waiting time until a buyer receives his ticket, or equivalently, the expected total waiting time for all buyers. In this situation, it is probably not reasonable to expend much effort in attaining a near-optimal matching, for the time to find the matching can easily exceed the eventual savings in locating a nearby ticket. Even so, the expected total distance in an optimal matching is significant as a lower bound for the expected total waiting time.

This paper investigates properties of optimal resource placements and provides fast algorithms for finding them. It also provides upper bounds for the costs of optimal placements and of certain nearly-optimal and easily-described placements. The network in all cases is assumed to be configured as a tree. Nevertheless, the upper bounds can often be applied to an arbitrary (connected) distributed network by constructing a spanning tree.

In this paper, we allow for the possibility that fractions of resources, and not only whole resources, might be located at some nodes of the network tree. This is reasonable if, for instance, the resources are large blocks of available data storage space. It would be perfectly permissable for a user to obtain parts of his needed storage from several different nodes. On the other hand, we assume that the requests are discrete -- each request is for one unit of resource.

In Section 2, we present our notation, definitions, and those results which apply to arbitrary trees and arbitrary probability distributions for arrivals of requests. §2.1 defines matchings and relates them to network flows. §2.2 defines the expected total cost of a placement in terms of expected total flow. Both the function describing the expected flow on each edge and the function describing the minimum possible expected total flow for any subtree are of a particularly simple form -- they are unbounded, convex, piecewise linear functions on the nonnegative reals, with

[†]This research was supported in part by the National Science Foundation under grants MCS77-02474, MCS77-15628, MCS78-01698, MCS80-03337, U.S. Army Research Office Contract Number DAAG29-79-C-0155, and the Office of Naval Research Contract Numbers N00014-79-C-0873 and N00014-80-C-0221.

all singularities at integers. This immediately implies (in §2.3) that there are always optimal resource placements consisting of whole numbers of resources located at each node; it is never necessary to place fractions of resources at any node in order to achieve an optimal placement. An algorithm using $O(t^2 \cdot \#\text{edges})$ arithmetic operations is presented which always finds an optimal whole resource placement. This is the fastest algorithm we have which is completely general. Section 4 contains faster algorithms for special cases.

§2.4 considers what is required for a placement to optimize the expected flow over any particular edge in the tree -- i.e. to be locally optimal. Unfortunately, it is not possible in general to obtain a single placement which simultaneously optimizes the expected flow on all edges.

A fair whole placement is one which "almost optimizes" the flow on each edge -- that is, one in which the number of resources in each subtree is either the mean rounded up or down. It is always possible (and quite easy) to obtain a fair whole placement, and such placements are close to optimal.

In §2.5, we show that if there is a probability less than ½ that any request will arrive in a particular subtree, then it is always bad to place even a very small fraction of a resource anywhere in that subtree.

In Section 3, we analyze the cost of an optimal placement for the case of a complete binary tree of n leaves but with an arbitrary probability distribution for request arrivals. An arbitrary fair whole resource placement has expected total cost at most $O(n + \sqrt{t}\,\sqrt{n})$. Note that in the very important case where t is roughly proportional to n (i.e. the number of resources in the network is proportional to the number of nodes), that the expected average cost per request is bounded by a constant, independent of the size of the network. This situation is very different from the case of centralizing the resources, where the average cost grows proportionately with the log of the number of nodes in the network.

2. NOTATION, DEFINITIONS AND GENERAL RESULTS

2.1. Trees, Matchings and Flows

A tree $T = (V,E)$ is an undirected acyclic graph, where V is the vertex (node) set and E is the edge set.

Let N denote the natural numbers, including 0, and let R^+ denote the nonnegative reals.

A set of requests is described by a function $r: V \to R^+$ such that $r(v)$ is the number of requests originating at node v. A placement of resources is described by a function $s: V \to R^+$ such that $s(v)$ is the number of resources placed at node v. We always assume $\text{total}(r) = \text{total}(s)$, where for any $g: V \to R^+$, $\text{total}(g) = \sum_{v \in V} g(v)$.

A matching is a function $m: V \times V \to R^+$. $m(u,v)$ gives the number of requests at node u which are satisfied by resources at node v. m is exact for r,s if for all $u,v \in V$, $\sum_{w \in V} m(u,w) = r(u)$ and $\sum_{w \in V} m(w,v) = s(v)$. Thus, an exact matching gives a complete correspondence between requests and resources. Clearly, an exact matching exists whenever $\text{total}(r) = \text{total}(s)$.

The cost of a matching is defined to be

$$\text{cost}(m) = \sum_{u,v \in V} m(u,v) \cdot d(u,v)$$

where $d(u,v)$ is the number of edges in the unique path from u to v in T. Let

$$\text{cost}(r,s) = \min \{\text{cost}(m) \mid m \text{ is an exact matching for } r,s\}.$$

A matching m is optimal for r,s if $\text{cost}(m) = \text{cost}(r,s)$.

It is convenient to relate a matching to a flow in a directed graph. Choose a node w, and direct the edges of the tree to point away from w. Corresponding to the usual way of drawing trees with the root at the top, we let $\text{high}(e)$ be the endpoint of edge e which is closest to w, and $\text{low}(e)$ be the endpoint farthest from w.

In the remainder of this paper, it will be convenient for w itself to have an edge entering it. Therefore, we add a new node v to serve as the root, and we let $\text{rootedge}(T) = (v,w)$, an edge from v to w. Resources will never be placed on v, and requests will never originate at v; v and $\text{rootedge}(T)$ are just notational conveniences.

A flow is a function $f: E \to R$ which describes the "movement" of resources from their initial placement to corresponding requests. A positive value of $f(e)$ denotes a flow along the direction of the edge (i.e. towards the leaves) whereas a negative value of $f(e)$ denotes a flow towards the root. A flow is stable for r,s if for every $v \in V$,

$$r(v) + \sum_{e:\text{high}(e)=v} f(e) = s(v) + \sum_{e':\text{low}(e')=v} f(e').$$

Thus, at every node, the flow out equals the flow in.

The cost of a flow is defined to be

$$\text{cost}(f) = \sum_{e \in E} |f(e)|.$$

Call a flow f optimal for r,s if it is stable for r,s and has minimal cost over all such flows.

The following theorem formalizes the intuition the flows correspond to resources moving along edges and that in an optimal matching, resources do not move both directions along the same edge. We omit the straightforward but tedious proof.

Theorem 2.1.1. Let m be an optimal matching and f an optimal flow for r,s. Then $\mathrm{cost}(m) = \mathrm{cost}(f)$. Moreover, if either m or f has an integral range, then the other can be chosen so also.

The removal of any edge e of a tree T splits the tree into two disconnected components: $B(e)$, the nodes "below" e, and $A(e)$, the nodes "above" e. $B(e)$ and $A(e)$ are defined by:

$$B(e) = \{v \in V \mid v \text{ is in the subtree rooted by } \mathrm{low}(e)\}$$

$$A(e) = V - B(e).$$

The flow along e in any flow for r,s depends only on the total number of requests and the total number of resources in $B(e)$. For any function $g\colon V \to R^+$ and $e \in E$, let

$$\underline{\mathrm{total}}(e,g) = \sum_{v \in B(e)} g(v).$$

If $\mathrm{total}(e,r)$ exceeds $\mathrm{total}(e,s)$, then there are more requests than resources in the subtree $B(e)$, so the excess must be satisfied by resources flowing into $B(e)$ via the edge e. On the other hand, if $\mathrm{total}(e,s)$ exceeds $\mathrm{total}(e,r)$, the excess resources in $B(e)$ are needed by requests in $A(e)$ (since $\mathrm{total}(r) = \mathrm{total}(s)$), so they must flow out of the subtree via e. By our convention that the sign of the flow denotes its direction, the directed flow in either case is given by

$$\mathrm{dflow}_{r,s}(e) = \mathrm{total}(e,r) - \mathrm{total}(e,s).$$

Theorem 2.1.2. $\mathrm{dflow}_{r,s}$ is the unique stable flow for r,s.

Proof. The argument sketched above shows that there is at most one stable flow for r,s. We leave to the reader to show that $\mathrm{dflow}_{r,s}$ is stable for r,s. []

Theorems 2.1.1 and 2.1.2 immediately yield:

Theorem 2.1.3. $\mathrm{cost}(r,s) = \mathrm{cost}(\mathrm{dflow}_{r,s})$.

Thus, our original problem of studying optimal exact matchings reduces to the problem of studying a particular stable flow.

2.2. Expected Costs

In this section, we introduce probability distributions for sets of requests and define costs of placements in terms of their expected costs given a randomly chosen set of requests.

If ϕ is a probability function on V and $t \in N$, then a random $r\colon V \to R^+$ can be chosen according to a probability distribution determined as follows: for each i in turn, $1 \le i \le t$, ϕ is used to select a vertex. Then $r(v)$ is the total number of times v is selected for each $v \in V$. We always assume $\phi(\mathrm{root}) = 0$.

Fix ϕ, t as above, and let $s\colon V \to R^+$ with $\mathrm{total}(s) = t$. Then $\underline{\mathrm{expcost}}(s)$ denotes the expected value of $\mathrm{cost}(r,s)$, where r is chosen as described above, and $\underline{\mathrm{minexpcost}} =$ $\min\{\mathrm{expcost}(s) \mid s\colon V \to R^+ \text{ and } \mathrm{total}(s) = t\}$.

The flow along an edge e depends only on $\mathrm{total}(e,s)$, the number resources in the subtree below e, and not on their particular placement. Let $\underline{\mathrm{expflow}_e}(u)$ be the expected value of $|\mathrm{dflow}_{r,s}(e)|$, where r is chosen randomly as described above, and s is any placement with $\mathrm{total}(e,s) = u$.

The following two expansions are easy to see.

Theorem 2.2.1. $\mathrm{Expcost}(s) = \sum_{e \in E} \mathrm{expflow}_e(\mathrm{total}(e,s))$.

Theorem 2.2.2. $\mathrm{Expflow}_e(u) = \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |u-i|$, where $p = \sum_{v \in B(e)} \phi(v)$.

The next theorem shows that the expflow function has a simple form.

Theorem 2.2.3. For any fixed ϕ, $t \ge 1$, and $e \in E$, $\mathrm{expflow}_e$ is an unbounded, convex, piecewise linear function from R^+ to R^+, with all singularities occurring at integer values.

Proof. By Theorem 2.2.2, $\mathrm{expflow}_e(u)$ is the sum of functions $g_i(u) = k_i \cdot |u-i|$, where k_i does not depend on u, $0 \le i \le t$. Each g_i is a convex, piecewise linear function from R^+ to R^+ with a singularity at $u = i$, and at least one of the g_i's is unbounded. Since addition preserves all four required properties, the result follows.

If $e \in E$, let E_e denote the set consisting of e, together with all edges below e in T, so $d \in E_e$ if $\mathrm{low}(d) \in B(e)$. Define

$$\underline{\mathrm{expcost}_e}(s) = \sum_{d \in E_e} \mathrm{expflow}_d(\mathrm{total}(d,s))$$

and

$$\underline{\mathrm{minexpcost}_e}(u) = \min\{\mathrm{expcost}_e(s) \mid \mathrm{total}(e,s) = u\}.$$

Thus, $\text{expcost}_e(s)$ gives the portion of the expected cost of placement s due to flow in the subtree below (and including) e, and $\text{minexpcost}_e(u)$ *gives the least such cost, subject* to the constraint that exactly u resources be placed in the subtree.

From these definitions and Theorem 2.2.1, it should be clear that

$$\text{expcost}_e(s) = \text{expflow}_e(\text{total}(e,s)) + \sum_{i=1}^{\ell} \text{expcost}_{e_i}(s)$$

where $e_1,\dots,e_\ell$ are the immediate descendants of edge e. Optimizing over all s with $\text{total}(e,s) = u$, we get

Theorem 2.2.4. Let ϕ, t, and e be fixed. Let $e_1,\dots,e_\ell$ be the immediate descendant edges of e, and let $u \in R^+$. Then

$$\text{minexpcost}_e(u) = \text{expflow}_e(u) + \min\{\sum_{i=1}^{\ell} \text{minexpcost}_{e_i}(u_i) \mid \Sigma u_i \le u\}.$$

In order to obtain more information about the values of the minexpcost_e function, we first show that it has a simple form.

Theorem 2.2.5. For any fixed ϕ, $t \ge 1$, and e, minexpcost_e is an unbounded, convex, piecewise linear function from R^+ to R^+, with all singularities occurring at integer values.

Proof. We use induction on edges in the tree, working from the leaves toward the root.

If e is a lowest edge, then there is only one edge, e, in E_e. Thus, $\text{minexpcost}_e = \text{expflow}_e$, which has the needed properties by Theorem 2.2.3.

Now assume the result holds for edges below e, and let $e_1,\dots,e_\ell$ denote the immediate descendant edges of e. Consider the expression for $\text{minexpcost}_e(u)$ given in Theorem 2.2.4. The first term has the needed properties by Theorem 2.2.3. It remains to show that the second term is convex, piecewise linear and has all singularities at integers.

Write f_i for minexpcost_{e_i}, $g(u)$ for $\min \{ \Sigma f_i(u_i) \mid \Sigma u_i \le u \}$.

By the induction hypothesis, each f_i is unbounded, convex and piecewise linear with all singularities at integers. Hence, the derivative $f_i'(x)$ is defined at all non-singular points. We define $f_i'(n)$ for n a singularity of f_i to be the limit of $f_i'(x)$ as x approaches n from above. By convexity and linearity of f_i, f_i' is a non-decreasing step function over the domain R^+, with steps occurring at integer points.

For each i, $1 \le i \le \ell$, let r_i be the smallest element of R^+ with $f_i'(r_i) \ge 0$. r_i exists since f_i is unbounded, and $r_i \in N$ by the properties of f_i'. We consider two cases:

Case 1. $u \ge \Sigma r_i$. Then $g(u) = \Sigma f_i(r_i)$, so g is constant for all such u.

Case 2. $u < \Sigma r_i$. Let $S = \{f_i'(x) \mid 1 \le i \le \ell, x \in [0, r_i)\}$. For $\sigma \in S$ and $1 \le i \le \ell$, define

$$x_i^\sigma = \text{glb}\ \{z \mid f_i'(z) \ge \sigma\}$$

$$y_i^\sigma = \text{lub}\ \{z \mid f_i'(z) \le \sigma\}$$

By the properties of f_i', we have x_i^σ, $y_i^\sigma \in N$, $x_i^\sigma \le y_i^\sigma \le r_i$, and $f_i'(z) = \sigma$ iff $z \in [x_i^\sigma, y_i^\sigma)$. Hence, the intervals $[x_i^\sigma, y_i^\sigma)$ for $\sigma \in S$ partition $[0, r_i)$, so the intervals $I_\sigma = [\sum_i x_i^\sigma, \sum_i y_i^\sigma)$ for $\sigma \in S$ partition $[0, \Sigma r_i)$.

Choose $\sigma \in S$. We now show that g is linear over I_σ. Let $u \in I_\sigma$. Choose $\underset{\sim}{u} = (u_1,\dots,u_\ell)$ such that $g(u) = \Sigma f_i(u_i)$ and $\Sigma u_i \le u$. By the choice of u and using the facts that $f_i'(z) < \sigma$ iff $z < x_i^\sigma$ and $f_i'(z) > \sigma$ iff $z \ge y_i^\sigma$, it is straightforward to show that $x_i^\sigma \le u_i \le y_i^\sigma$ and $\Sigma u_i = u$. Hence, $f_i(u_i) = f_i^\sigma(x_i) + \sigma\cdot(u_i - x_i^\sigma)$. Summing over all i, we get

$$g(u) = \sum_{i=1}^{\ell} f_i(u_i)$$

$$= \Sigma f_i(x_i^\sigma) + \sigma\cdot(\Sigma u_i - \Sigma x_i^\sigma)$$

$$= \Sigma f_i(x_i^\sigma) + \sigma\cdot(u - \Sigma x_i^\sigma),$$

a linear function of u as desired.

The convexity of g follows from the monotinicity of the f_i'. □

Examination of the proof of Theorem 2.2.5 allows us to sharpen Theorem 2.2.4 by stating that *the minimum cost can always be achieved by placing* whole resources on all vertices.

Theorem 2.2.6. Let ϕ, t, e be fixed. Let $e_1,\dots,e_\ell$ be the immediate descendant edges of e, and let $u \in N$. Then $\text{minexpcost}_e(u) = \text{expflow}_e(u) + \min\{\sum_{i=1}^{\ell} \text{minexpcost}_{e_i}(u_i) \mid \Sigma u_i \le u \text{ and } u_i \in N\}$.

2.3. Optimal Placement

We are interested in determining the "best" placement functions $s: V \to R^+$ in the following sense. We say $s: V \to R^+$ is optimal for ϕ, t provided $\text{total}(s) = t$ and $\text{expcost}(s) = \text{minexpcost}$. The first characterization result follows immediately from Theorem 2.2.6 and shows that there are optimal s which take on integral values only.

Theorem 2.3.1. For any probability function ϕ and any $t \in N$, there exists $s: V \to N$ which is optimal for ϕ, t.

Proof. Theorem 2.2.6 essentially provides an algorithm for producing such s. For any edge e of T with immediate descendants $e_1,\dots,e_\ell$, and any $u \in N$, one determines values of s for all nodes below e by considering all possible decompositions $u_1,\dots,u_\ell$ with $\Sigma u_i \le u$ and $u_i \in N$ for all i. For each such decomposition, one recursively determines values of s for all nodes below each e_i, and corresponding costs. The decomposition with the smallest total cost is chosen. □

In order to analyze the cost of determining an optimal placement as above, we do not perform a straightforward recursive analysis of the algorithm described in the proof of Theorem 2.3.1. Rather, we take advantage of repeated work in various recursive calls. During the algorithms, one must calculate $\text{expflow}_e(u)$ for all $e \in E$ and all u, $0 \le u \le t$. The number of arithmetic operations involved in one calculation of $\text{expflow}_e(u)$ is $O(t)$ (if performed judiciously), independent of e and u. It is these costs which dominate the total count of arithmetic operations, so that an $O(t^2 \cdot |E|)$ analysis results. We summarize this discussion in the following theorem.

Theorem 2.3.2. There is an algorithm using $O(t^2 \cdot |E|)$ arithmetic operations which, for any tree $T = (E,V)$ probability function ϕ, and $t \in N$, determines a placement of whole resources which is optimal for ϕ, t.

2.4. Optimizing Flow on Individual Edges

In this section, we show that the flow on each individual edge is optimized for a number equal to the median of an appropriate binomial distribution. We use this to bound the distance from optimal of two simple placements.

Let $n \in N - \{0\}$, $0 \le p \le 1$. Let x be a random variable whose value is the number of successes in n independent trials, each of whose probability of success is p. Define median(n,p) as the smallest $c \in R^+$ such that $\Pr[x \le c] \ge \frac{1}{2}$.

Theorem 2.4.1. Let ϕ be a probability function on V, $t \in N - \{0\}$, $e \in E$. Then $\text{expflow}_e(u)$ is minimized at $u = \text{median}(t,p)$, where $p = \sum_{v \in B(e)} \phi(v)$.

Proof. Write f for expflow_e. By Theorem 2.2.3, f is minimized at an integer u which is the smallest $r \in N$ with $f(r+1) - f(r)$ nonnegative. Now

$$f(r+1) - f(r)$$

$$= \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |r+1-i|$$

$$- \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |r-i|$$

by Theorem 2.2.2,

$$= \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} (|r+1-i| - |r-i|)$$

$$= \sum_{i=0}^{r} \binom{t}{i} p^i (1-p)^{t-i} - \sum_{i=r+1}^{t} \binom{t}{i} p^i (1-p)^{t-i}$$

$$= 2 \sum_{i=0}^{r} \binom{t}{i} p^i (1-p)^{t-i} - 1$$

$$= 2\Pr[x \le r] - 1.$$

Thus, u is the smallest $r \in N$ with $2\Pr[x \le r] - 1$ nonnegative, or $\Pr[x \le r] \ge \frac{1}{2}$; that is, $u = \text{median}(t,p)$. □

The following theorem follows immediately from Theorem 3.2 and Corallary 3.1 of Jogdeo and Samuels [1]. It is also implicit in some earlier work by Uhlmann [2,3].

Theorem 2.4.2 (Jogdeo and Samuels). Let $n \in N$, $n \ge 1$, $0 \le p \le 1$. Then $\text{median}(n,p) \in \{\lfloor np \rfloor, \lceil np \rceil\}$.

Since np is the mean number of successes in n independent trials with success probability p, this theorem implies that the mean and median differ by less than one.

Thus, each edge individually has its flow optimized at a value which is either the mean rounded up or the mean rounded down. However, it is not always possible to achieve this local optimum consistently throughout the tree. In fact, in this very general setting, an optimal placement might require some subtree to contain a value other than the mean rounded up or down.

For any T, ϕ, t, and for each edge $e \in E$, let $p_e = \sum_{v \in B(e)} \phi(v)$. $s: V \to R^+$ is an <u>exact fair-share placement</u> for T, ϕ, t if for each $e \in E$, $\text{total}(e,s) = tp_e$. $s': V \to N$ is a <u>fair whole placement</u> for T, ϕ, t if for each $e \in E$, $\lfloor tp_e \rfloor \le \text{total}(e,s) \le \lceil tp_e \rceil$.

It is not difficult to see that for any T, ϕ, t, an exact fair-share placement and a fair whole placement exist. Let $s(v) = t \cdot \phi(v)$ for $v \in V$. Then s is an exact fair-share placement. Let $s'(v) = \lfloor t \cdot \phi(v) \rfloor$ if v is a leaf. Let $s'(v) = 0$ if v is the root. For v not a leaf or root, let e be the edge with $\text{low}(e) = v$, and let $e_1, \ldots, e_\ell$ be its immediate descendants. Let $s'(v) = \lfloor tp_e \rfloor - \sum_{i=1}^{\ell} \lfloor tp_{e_i} \rfloor$. An easy induction shows that s' is a fair whole placement for T, ϕ, t, and in fact, $\text{total}(e,s') = \lfloor tp_e \rfloor$ for every $e \in E$.

In the special case that ϕ is non-zero only on leaves, then there is a fair whole placement s'' which is non-zero only on leaves. It can be obtained by a simple recursive construction. Start at the root of T with t resources to distribute. Below any edge e in the tree, we will have either $\lfloor tp_e \rfloor$ or $\lceil tp_e \rceil$ to distribute. Assume e has descendants $e_1, \ldots e_\ell$. Distribute $\lfloor tp_{e_i} \rfloor$ or $\lceil tp_{e_i} \rceil$ to the i^{th} subtree. Since

$$\sum_{i=1}^{\ell} \lfloor tp_{e_i} \rfloor \le \lfloor tp_e \rfloor \le \lceil tp_e \rceil \le \sum_{i=1}^{\ell} \lceil tp_{e_i} \rceil$$

the distribution is possible. We then proceed recursively on $e_1, \ldots, e_\ell$.

<u>Example 2.4.1</u>. Let T be a 32-leaf complete binary tree (except for rootedge(T)), $\phi(\ell) = \frac{15}{256}$ for each of the leftmost 16 leaves ℓ, $\frac{1}{256}$ for each of the rightmost 16 leaves, and 0 for all other nodes. Let $t = 16$. The placement s which has $s(\ell) = 1$ for each of the leftmost 16 leaves ℓ, and 0 elsewhere, has $\text{total}(e,s) = 16$ for e the left descendant of rootedge(T). However, the mean number of requests in the subtree below e is $t \cdot \frac{15}{16} = 15$, so s is not a fair whole placement. Nevertheless, one can determine by exhaustive searching that s is optimal, and *expcost(s) is strictly smaller than expcost(s')* for any fair whole placement s'.

We note in general, however, that the optimal cost cannot be <u>too</u> much less than the cost of any exact fair-share placement or fair whole placement.

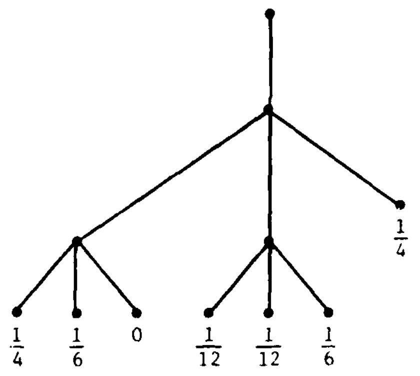
<u>Theorem 2.4.3</u>. *Let s be an exact fair-share placement or a fair whole placement for T, ϕ, t. Then $\text{expcost}(s) \le \text{minexpcost} + |E|$.*

<u>Proof</u>. By the results of this section, expflow_e is minimized at some u, where $\lfloor tp_e \rfloor \le u \le \lceil tp_e \rceil$. Thus, $|\text{total}(e,s) - u| \le 1$. But then $|\text{expflow}_e(\text{total}(e,s)) - \text{expflow}_e(u)| \le 1$, by calculations similar to those in the proof of Theorem 2.4.1. Since no placement can do better than the optimal on each edge, s incurs at most an extra cost of 1 per edge. $\square$

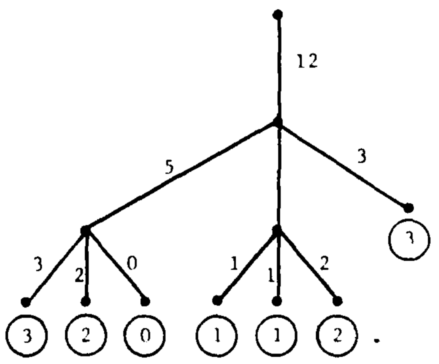
For some choices of T, t and ϕ, of course, it is possible to consistently achieve the median on each edge -- for example, if the mean is an integer for every edge. But in general, we have no global optimality results. In §4 we obtain optimal results for a special case.

<u>Example 2.4.2</u>. If T is a complete binary tree with leaf vertices L, $|L| = 2^k$, $t = a \cdot 2^k$ for integer a, and ϕ the uniform distribution on the leaves, then the placement s such that $s(v) = a$ for each $v \in L$ and $s(v) = 0$ for $v \in V - L$ is optimal, because it achieves the integer mean on each edge.

<u>Example 2.4.3</u>. Let T be the tree depicted below, ϕ as indicated on the leaves and 0 elsewhere, and $t = 12$.



Then the means are indicated on each edge below, so all resources can be placed at the levels indicated by the circled numbers.



2.5. Nodes with Zero Placements

We conclude this section with a somewhat surprising characterization theorem. It says that if there is a probability less than $\frac{1}{2}$ of any request arriving in a subtree, then it is bad to place even a small fraction of a resource anywhere in that subtree.

Theorem 2.5.1. Let $e \in E$ satisfy $(1-p)^t > \frac{1}{2}$, where $p = \sum_{v \in B(e)} \phi(v)$. Let s be optimal for ϕ, t. Then $s(v) = 0$ for all $v \in B(e)$.

Proof. Assume not, and fix e, s exhibiting the contrary. Choose e' below e to be a lowest edge for which $x = s(\text{low}(e')) > 0$. Define a new placement s', where $s'(\text{low}(e')) = 0$, $s'(\text{high}(e') = s(\text{high}(e')) + x$, and $s'(v) = s(v)$ otherwise. We show that $\text{expcost}(s') < \text{expcost}(s)$, contradicting the optimality of s.

By Theorem 2.2.1, e' is the only edge for which s and s' have different expected absolute flows, so

$$\text{expcost}(s) - \text{expcost}(s')$$

$$= \text{expflow}_{e'}(\text{total}(e',s)) - \text{expflow}_{e'}(\text{total}(e',s'))$$

$$= \text{expflow}_{e'}(x) - \text{expflow}_{e'}(0).$$

Let $r = \sum_{v \in B(e')} \phi(v)$. Applying Theorem 2.2.2, the difference is

$$\sum_{i=0}^{t} \binom{t}{i} r^i (1-r)^{t-i} (|x-i| - i)$$

$$\geq x(1-r)^t + \sum_{i=1}^{t} \binom{t}{i} r^i (1-r)^{t-i} (-x)$$

$$= x(2(1-r)^t - 1).$$

Since $(1-r)^t \geq (1-p)^t > \frac{1}{2}$, the difference > 0, giving the needed contradiction. □

3. BOUNDS ON THE COSTS OF OPTIMAL PLACEMENTS

In this section, we assume that T is a complete (balanced) binary tree (except for rootedge(T)) with leaves L, and let $n = |L|$. Assume ϕ is arbitrary, except that as always $\phi(\text{root}) = 0$. We show that optimal placements are much better than centralized placements; in fact, their expected cost is linear in the number of leaves of the tree.

3.1. Cost of Exact Fair-Share Placements

Since we do not have a direct characterization of optimal placements, we instead bound the expected cost of an arbitrary exact fair-share placement. This upper bound, of course, provides an upper bound for optimal placements. Moreover, by Theorem 2.4.3, the costs cannot differ by more than $2n$.

Theorem 3.1.1. Let T, ϕ, L, n be as above. Let $t \in N$, $t \geq 1$, and let $s: V \to R^+$ be an exact fair-share placement with $\text{total}(s) = t$. Let $e \in E$ and $p_e = \sum_{v \in B(e)} \phi(v)$, $m = |L \cap B(e)|$. Then

$$\text{expcost}_e(s) \leq c\sqrt{mtp_e}$$

Here c is a fixed constant independent of the choice of T, ϕ, t, s, or e.

Before proving the theorem, we need some technical lemmas.

Lemma 3.1.2. For $t \in N$, $t \geq 1$, $s \leq t-1$, $0 \leq p \leq 1$, it is the case that

$$\sum_{i=0}^{s} \binom{t}{i} p^i (1-p)^{t-i} (tp-i) = tp\binom{t-1}{s} p^s (1-p)^{t-s}.$$

Proof. $\binom{t}{i} p^i (1-p)^{t-i} (tp-i)$

$$= tp\binom{t}{i} p^i (1-p)^{t-i} - t\binom{t-1}{i-1} p^i (1-p)^{t-i}.$$

Since $\binom{t}{i} = \binom{t-1}{i} + \binom{t-1}{i-1}$, this expression is in turn equal to

$$tp\binom{t-1}{i} p^i (1-p)^{t-i} + tp\binom{t-1}{i-1} p^i (1-p)^{t-i}$$

$$- t\binom{t-1}{i-1} p^i (1-p)^{t-i}$$

$$= tp\binom{t-1}{i} p^i (1-p)^{t-i} - tp\binom{t-1}{i-1} p^{i-1} (1-p)^{t-(i-1)}.$$

Thus $\sum_{i=0}^{s} \binom{t}{i} p^i (1-p)^{t-i} (tp-i)$

$$= \binom{t}{0}(1-p)^t tp + tp\left[\sum_{i=1}^{s} \binom{t-1}{i} p^i (1-p)^{t-i} - \sum_{i=1}^{s} \binom{t-1}{i-1} p^{i-1} (1-p)^{t-(i-1)}\right]$$

$$= tp(1-p)^t + tp\left[\sum_{i=1}^{s} \binom{t-1}{i} p^i (1-p)^{t-i} - \sum_{i=0}^{s-1} \binom{t-1}{i} p^i (1-p)^{t-i}\right]$$

$$= tp(1-p)^t + tp\left[\binom{t-1}{s} p^s (1-p)^{t-s} - \binom{t-1}{0} p^0 (1-p)^t\right]$$

$$= tp\binom{t-1}{s} p^s (1-p)^{t-s}.$$

Lemma 3.1.3. If $t \in N$, $t \geq 1$, $0 \leq p \leq 1$, then

$$\sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |tp-i| = 2tp\binom{t-1}{\lfloor tp \rfloor} p^{\lfloor tp \rfloor} (1-p)^{t-\lfloor tp \rfloor}.$$

Proof. $\sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |tp-i|$

$$= \sum_{i=0}^{\lfloor tp \rfloor} \binom{t}{i} p^i (1-p)^{t-i} (tp-i)$$

$$- \sum_{i=\lfloor tp \rfloor +1}^{t} \binom{t}{i} p^i (1-p)^{t-i} (tp-i)$$

$$= 2 \sum_{i=0}^{\lfloor tp \rfloor} \binom{t}{i} p^i (1-p)^{t-i} (tp-i)$$

$$- \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} (tp-i)$$

$$= 2tp \binom{t-1}{\lfloor tp \rfloor} p^{\lfloor tp \rfloor} (1-p)^{t-\lfloor tp \rfloor}$$

$$- \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} (tp-i)$$

by Lemma 3.1.2. But

$$\sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} (tp-i)$$

$$= \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} tp - \sum_{i=1}^{t} \binom{t}{i} p^i (1-p)^{t-i} i$$

$$= tp \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i}$$

$$- tp \sum_{i=1}^{t} \binom{t-1}{i-1} p^{i-1} (1-p)^{(t-1)-(i-1)} = tp - tp = 0 .$$

□

Lemma 3.1.4. For $t \in N$, $t \geq 1$, $0 \leq p < 1$, $tp \geq 1$, it is the case that

$$\binom{t-1}{\lfloor tp \rfloor} (1-p)^{t-\lfloor tp \rfloor} p^{\lfloor tp \rfloor} \leq (1 + \frac{1}{4t}) \cdot \frac{e}{\sqrt{2\pi \lfloor tp \rfloor}}$$

Proof. A version of Stirling's formula says that for all $n \in N$, $n \geq 1$, it is the case that

$$(\frac{n}{e})^n \sqrt{2\pi n} \leq n! \leq (\frac{n}{e})^n \sqrt{2\pi n} \, (1 + \frac{1}{4n}) .$$

$tp < t$ so $\lfloor tp \rfloor \leq t - 1$ and $t - \lfloor tp \rfloor \geq 1$.

Then

$$\binom{t-1}{\lfloor tp \rfloor} = \frac{t-\lfloor tp \rfloor}{t} \cdot \binom{t}{\lfloor tp \rfloor}$$

$$\leq (\frac{t-\lfloor tp \rfloor}{\sqrt{t}}) \cdot (1 + \frac{1}{4t})$$

$$\cdot (\frac{t^t}{\sqrt{2\pi \lfloor tp \rfloor (t-\lfloor tp \rfloor)} \cdot \lfloor tp \rfloor^{\lfloor tp \rfloor} (t-\lfloor tp \rfloor)^{t-\lfloor tp \rfloor}}).$$

Therefore,

$$\binom{t-1}{\lfloor tp \rfloor} (1-p)^{t-\lfloor tp \rfloor} p^{\lfloor tp \rfloor}$$

$$\leq (\frac{t-\lfloor tp \rfloor}{\sqrt{t}}) \cdot (1 + \frac{1}{4t}) \cdot (\frac{t-tp}{t-\lfloor tp \rfloor})^{t-\lfloor tp \rfloor}$$

$$\cdot (\frac{tp}{\lfloor tp \rfloor})^{\lfloor tp \rfloor} \cdot (\frac{1}{\sqrt{2\pi \lfloor tp \rfloor (t-\lfloor tp \rfloor)}})$$

$$= \sqrt{\frac{t-\lfloor tp \rfloor}{2\pi t \lfloor tp \rfloor}} \cdot (1 + \frac{1}{4t}) \cdot (\frac{t-tp}{t-\lfloor tp \rfloor})^{t-\lfloor tp \rfloor} \cdot (\frac{tp}{\lfloor tp \rfloor})^{\lfloor tp \rfloor} .$$

But

$$(\frac{tp}{\lfloor tp \rfloor})^{\lfloor tp \rfloor} = (1 + \frac{tp-\lfloor tp \rfloor}{\lfloor tp \rfloor})^{\lfloor tp \rfloor} \leq (1 + \frac{1}{\lfloor tp \rfloor})^{\lfloor tp \rfloor}$$

$$\leq e, \quad \frac{t-\lfloor tp \rfloor}{t} \leq 1 \quad \text{and} \quad (\frac{t-tp}{t-\lfloor tp \rfloor})^{t-\lfloor tp \rfloor} \leq 1.$$

The lemma follows.

Lemma 3.1.5. Let T, ϕ, t be as in *Theorem 3.1.1*. Let $e \in E$ and let $p = \sum_{v \in B(e)} \phi(v)$. Then $\text{expflow}_e(tp) < 6 \cdot \sqrt{tp}$.

Proof. If $p = 0$ or $p = 1$, then $\text{expflow}_e(tp) = 0$. Assume $0 < p < 1$. By Theorem 2.2.2 and Lemma 3.1.3,

$$\text{expflow}_e(tp) = \sum_{i=0}^{t} \binom{t}{i} p^i (1-p)^{t-i} |tp-i|$$

$$= 2tp \binom{t-1}{\lfloor tp \rfloor} (1-p)^{t-\lfloor tp \rfloor} p^{\lfloor tp \rfloor}$$

If $tp \geq 1$, Lemma 3.1.4 shows this is at most

$$2tp \cdot (1 + \frac{1}{4t}) \frac{e}{\sqrt{2\pi \lfloor tp \rfloor}}$$

$$\leq 2 \cdot tp \cdot (\frac{5}{4}) \cdot \frac{e}{\sqrt{2\pi}} \cdot \frac{\sqrt{\lfloor tp \rfloor}}{\lfloor tp \rfloor}$$

$$\leq 2 \cdot 2 \cdot (\frac{5}{4}) \cdot \frac{e}{\sqrt{2\pi}} \cdot \sqrt{tp}$$

$$< 6 \cdot \sqrt{tp} .$$

On the other hand, if $tp < 1$, then

$$2tp \cdot \binom{t-1}{\lfloor tp \rfloor} (1-p)^{t-\lfloor tp \rfloor} p^{\lfloor tp \rfloor} = 2tp \cdot (1-p)^t < 2\sqrt{tp}.$$

In any case,

$$\text{expflow}_e(tp) < 6 \cdot \sqrt{tp} .$$

Proof of Theorem 3.1.1. Define a function $G: N \times R^+ \to R^+$ recursively:

$$G(0, u) = 6 \cdot \sqrt{u} \; ;$$

$$G(k, u) = \max \{G(k-1,u_1) + G(k-1,u_2) + 6 \cdot \sqrt{u} \mid u_1 + u_2 \le u\} \; , \quad k > 1 \; .$$

We first show by induction on m that $expcost_e(s) \le G(\log_2 m, tp_e)$. Since s is an exact fair-share placement, $total(e,s) = tp_e$.

$m = 1$: Then $low(e)$ is a leaf, so

$$\begin{aligned} expcost_e(s) &= expflow_e(total(e,s)) \\ &< 6 \cdot \sqrt{tp_e} \quad \text{by Lemma 3.1.5} \\ &= G(0, tp_e). \end{aligned}$$

$m > 1$: Then $m = 2^k$ since T is a complete binary tree. Let e_1, e_2 be the two immediate descendant edges of e. Then

$$expcost_e(s) = expflow_e(total(e,s)) + \sum_{i=1}^{2} expcost_{e_i}(s).$$

By induction,

$$expcost_{e_i}(s) \le G(k-1, tp_{e_i}), \quad i = 1, 2.$$

Again using Lemma 3.1.5, we have

$$\begin{aligned} expcost_e(s) &< 6 \cdot \sqrt{tp_e} + \sum_{i=1}^{2} G(k-1, tp_{e_i}) \\ &\le G(k, tp_e) = G(\log_2 m, tp_e) \end{aligned}$$

since $tp_{e_1} + tp_{e_2} \le tp_e$.

We complete the proof of the theorem by showing that

$$G(k, u) \le c \cdot 2^{k/2} \cdot \sqrt{u}$$

for some constant c.

First observe that for each $k \in N$, $G(k, u)$, regarded as a function of u, is concave and monotonically increasing. This is clearly true for $k = 0$. For $k > 0$, we know inductively that $G(k-1, u)$ is concave and monotonically increasing. Hence, $G(k-1,u_1) + G(k-1,u_2) \le 2 \cdot G(k-1, (u_1+u_2)/2) \le 2 \cdot G(k-1, u/2)$ whenever $u_1 + u_2 \le u$, so

(*) $G(k, u) = 2 \cdot G(k-1, u/2) + 6 \cdot \sqrt{u}$,
and $G(k, u)$ is concave and increasing.

We proceed to solve the recurrence equation (*). First, substitute $a \cdot 2^k$ for u in (*) to get

$$G(k, a \cdot 2^k) = 2 \cdot G(k-1, a \cdot 2^{k-1}) + 6 \cdot 2^{k/2} \cdot \sqrt{a}$$

Now, divide both sides by 2^k and express in terms of G', where $G'(k, a) = G(k, a \cdot 2^k)/2^k$, to get

$$G'(k, a) = G'(k-1, a) + \frac{6 \cdot \sqrt{a}}{2^{k/2}} \; .$$

Also,

$$G'(0, a) = G(0, a) = 6 \cdot \sqrt{a} \; .$$

Hence,

$$G'(k, a) = \sum_{i=0}^{k} \frac{6 \cdot \sqrt{a}}{2^{i/2}} \le 6 \cdot \sqrt{a} \sum_{i=0}^{\infty} 2^{-i/2} \; .$$

The series is convergent, so let $c = 6 \cdot \sum_{i=0}^{\infty} 2^{-i/2}$.

Then $G'(k,a) \le c\sqrt{a}$. Substituting back, we get

$$G(k, u) = 2^k \cdot G'(k, u/2^k) \le c \cdot 2^{k/2} \cdot \sqrt{u}$$

as desired.

We conclude that

$$expcost_e(s) \le G(\log_2 m, tp_e) \le c \cdot \sqrt{m tp_e} \; .$$ □

Corollary 3.1.6. Let T, ϕ, L, n, t be as in Theorem 3.1.1. Let $s: V \to R^+$ be an exact fair-share placement for T, ϕ, t, and let $s': V \to N$ be a fair whole placement for T, ϕ, t. Then

$$expcost(s) \le c \cdot \sqrt{nt}$$

and

$$expcost(s') \le c \cdot \sqrt{nt} + 2n - 1.$$

Proof. The first bound comes from a direct application of Theorem 3.1.1 to rootedge(T). The second bound follows from the first using Theorem 2.4.3 and that fact that an n-leaf binary tree has at most $2n-1$ edges (including rootedge(T)). □

3.2. Cost of Centralized Placement

s is a centralized placement if $s(v) = 0$ everywhere except at $low(rootedge(T))$, and $s(low(rootedge(T))) = t$. For such an s,

$$expcost(s) = t \log_2(n).$$

Note that the ratio of expected cost for a centralized placement to a fair whole placement is at least

$$\frac{t \log_2(n)}{c \sqrt{t} \sqrt{n} + 2n}$$

When t is small relative to n, then the centralized placement is superior, for reasons similar to those discussed in Section 2.5.

However, for $t = \Omega(n)$, the fair whole placement is better by a factor of $\Omega(\log_2 n)$, and for $t >> n$, the ratio approaches $\sqrt{t}$, that is, the centralized placement is worse by a square.

Note also that for $t = \Omega(n)$, the fair whole placement has linear expected cost, so the expected cost per request is a constant, whereas the cost per request for a centralized placement is $\log_2 n$.

4. OPTIMAL PLACEMENTS FOR SPECIAL DISTRIBUTION

In this section, we give several characterization results and algorithms for optimal placements for a further restriction of the case considered in Section 3 in which we assume ϕ is the same on all leaves and zero elsewhere. Specifically, we assume the following for this section:

(a) $T = (V, E)$ *is a complete binary tree* with leaves $L \subseteq V$, $n = |L|$, except that, as before, the root has a single emanating edge, rootedge(T).

(b) $\phi(v) = 1/n$ *for all* $v \in L$, *and* $\phi($ $= 0$ *for* all $v \in V - L$.

(c) $t \in N$, $t \geq 1$.

We define the level *of a vertex to be its* distance from the root. By our conventions regarding rootedge(T), the leaves are at level $\log_2 n + 1$.

For the special case being considered in this section, certain of the relevant definitions can be generalized to reflect the symmetry in the tree. For example, if e_1 and e_2 are edges with high(e_1) and high(e_2) at the same level k, then $\text{expflow}_{e_1} = \text{expflow}_{e_2}$. Therefore, we write expflow_k in place of either. Similarly, we write minexpcost_k for minexpcost_e, where k is the level of high(e).

4.1. A Bound on Levels with Nonzero Placements

Theorem 2.5.1 can be used to bound the level at which nonzero placement can occur in an optimal placement.

Theorem 4.1.1. Let $v \in V$ be at level $k \geq \lceil \log_2 t \rceil + 2$. Let s be optimal for ϕ, t. Then $s(v) = 0$.

Proof. By Theorem 2.5.1, it suffices to show that $(1 - \frac{1}{2^{\lceil \log_2 t + 1 \rceil}})^t > \frac{1}{2}$ for $t \in N$. It also suffices to consider t a power of 2. In this case, the inequality is just $(1 - \frac{1}{2t})^t > \frac{1}{2}$, which follows by an easy induction on $\log_2 t$. □

Let $T_t = (V_t, E_t)$ denote the tree consisting of the vertices of T at levels from 0 to $\lceil \log_2 t \rceil + 1$ inclusive and the edges of T between them. The leaves L_t of T_t are the vertices at level $\lceil \log_2 t \rceil + 1$, and $n_t = |L_t|$. Let ϕ_t be defined on the vertices of T_t by $\phi_t(v) = 1/n_t$ for all $v \in L_t$ and $\phi_t(v) = 0$ for $v \in V_t - L_t$. If $s: V \to R^+$ is such that $s(v) = 0$ for all $v \in V$ at levels below $\lceil \log_2 t \rceil + 1$, then $s_t: V_t \to R^+$ can be defined from s by simply ignoring the missing vertices. Then it is easy to see that $\text{expcost}^{(T)}(s) = \text{expcost}^{(T_t)}(s_t) + t \cdot (\log_2 n - \lceil \log_2 t \rceil)$. (Superscripts distinguish the trees under consideration.) We then have the following.

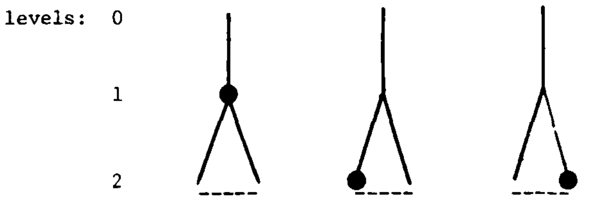
Theorem 4.1.2. If $t > 1$, then $\text{minexpcost}^{(T)} = \text{minexpcost}^{(T_t)} + t \cdot (\log_2 n - \lceil \log_2 t \rceil)$.

Proof. ($\geq$) follows from Theorem 4.1.1 and the remarks.

($\leq$) follows because any placement for T_t can be augmented to a placement for T by placing zeros on the additional nodes, thereby incurring the stated cost. □

Thus, in the remainder of this section, we assume that the maximum level in T is at most $\lceil \log_2 t \rceil + 1$ (that is, $n < 2t$). The reader can then use Theorem 4.1.2 to infer corresponding results about cases where $n \geq 2t$.

Example 4.1.1. The case where $t = 1$ is somewhat peculiar. The reader can verify that for all T with maximum level number at least 2, the following pictures all represent optimal placements.



That is, in the first case,

$$s(v) = \begin{cases} 1 & \text{for } v \text{ the son of the root,} \\ 0 & \text{otherwise,} \end{cases}$$

while in the other two cases,

$$s(v) = \begin{cases} 1 & \text{for } v \text{ the left (resp. right) grandson of the root,} \\ 0 & \text{otherwise.} \end{cases}$$

This is the only value of t for which an optimal placement can have nonzero values below level $\lceil \log t \rceil + 1$.

Example 4.1.2. If $t = 2^k$ and $n > t$, then the placement with one resource on each of the t vertices at level $k+1$ is optimal: nothing is placed below this level, and an optimal placement for the whole tree results from an optimal placement within levels up to $\log t + 1$. But clearly putting one resource across all leaves of T_t is optimal, as seen in Example 2.4.2.

4.2. Algorithms for Finding Optimal Placements

In this section, we prove two sharper versions of Theorem 2.2.4 for the special case of this section, and use them as bases for two algorithms for finding optimal placements.

Theorem 4.2.1. Let $t \in N$, $u \in R^+$ and $k \in N$, $k \leq \log_2 n - 1$. Then $\text{minexpcost}_k(u) = \text{expflow}_k(u)$ $+ 2 \cdot \min\{\text{minexpcost}_{k+1}(u') \mid u' \in \{0,1,\ldots,\lfloor \frac{u}{2} \rfloor, \frac{u}{2}\}\}$.

Proof. $\leq$ is clear. We show $\geq$. Write f for minexpcost_{k+1}. Consider any u_1, $u_2 \in R^+$ with $u_1 + u_2 \leq u$ and $f(u_1) + f(u_2)$ minimal. We will produce $u' \in \{0,1,\ldots,\lfloor \frac{u}{2} \rfloor, \frac{u}{2}\}$ with $2f(u')$ $\leq f(u_1) + f(u_2)$.

By Theorem 2.2.5, there exists $r \in N$ such that r is the smallest element of R^+ with $f(r) \leq f(s)$ for all $s \geq r$. We consider two cases.

Case 1. $u \geq 2r$.

Choose $u' = r$. $u' \in \{0,1,\ldots,\lfloor \frac{u}{2} \rfloor\}$ and $2f(u') \leq f(u_1) + f(u_2)$.

Case 2. $u < 2r$.

Then $u_1 + u_2 = u$. Choose $u' = \frac{u}{2}$. Then $2f(u') = 2f(\frac{u_1+u_2}{2}) \leq f(u_1) + f(u_2)$ by convexity of f.

An appeal to Theorem 2.2.4 completes the proof.

Theorem 4.2.2. For any $t \in N$, there exists s such that $s(v) = s(v')$ for all pairs of vertices v, v' at the same level, which is optimal for t.

Proof. s can be found by a recursive algorithm based on Theorem 4.2.1. □

We proceed as before to analyze the cost of finding the placement of Theorem 4.2.2. During the algorithm, one must calculate $\text{expflow}_0(t)$, $\text{expflow}_1(\frac{t}{2}),\ldots,\text{expflow}_{\log(n)}(\frac{t}{n})$. In addition, one must calculate $\text{expflow}_1(i)$ for all $i \in N$, $i \leq \frac{t}{2}$, $\text{expflow}_2(\frac{i}{2})$ for all $i \in N$, $i \leq \frac{t}{2}$, $\text{expflow}_3(\frac{i}{4})$ for all $i \in N$, $i \leq \frac{t}{2},\ldots$, $\text{expflow}_{\log(n)}$ $(\frac{2i}{n})$ for all $i \in N$, $i \leq \frac{t}{n}$.

This is a total of $O(t \log n)$ expflow computations. Since each such computation involves $O(t)$ arithmetic operations, we have the following theorem.

Theorem 4.2.3. There is an algorithm using $O(t^2 \log n)$ arithmetic operations which, for any tree T with n leaves satisfying the assumptions of this section and for any $t \in N$, determines an s which is optimal for t such that $s(v) = s(v')$ for all pairs of vertices v, v' at the same level.

Note that the bound of Theorem 4.2.3 represents an improvement over the bound in Theorem 2.3.2 applied to the special case of this section; the placements produced by the two algorithms have somewhat different properties, however. The remainder of this subsection deals with integral placements, the situation considered in Theorem 2.3.2.

Theorem 4.2.4. Let t, u, $k \in N$, $k \leq \log_2 n - 1$. Then

$$\text{minexpcost}_k(u) = \text{expflow}_k(u) + \min\{\text{minexpcost}_{k+1}(u_1) + \text{minexpcost}_{k+1}(u_2) \mid u_1, u_2 \in N, u_1 + u_2 \leq u, \text{ and } |u_2 - u_1| \leq 1\}.$$

Proof. Again, we show $\geq$. Write f for minexpcost_{k+1}. By Theorem 4.2.1, there is a value $u' \in \{0,1,\ldots,\lfloor \frac{u}{2} \rfloor, \frac{u}{2}\}$ such that $\text{minexpcost}_k(u) =$ $\text{expflow}_k(u) + 2f(u')$. If $u' \in \{0,\ldots,\lfloor \frac{u}{2} \rfloor\}$, then we simply take $u_1 = u_2 = u'$. If $u' = \frac{u}{2} \notin N$, then we take $u_1 = \lfloor \frac{u}{2} \rfloor$ and $u_2 = \lceil \frac{u}{2} \rceil$; in this case, piecewise linearity of f guarantees the required properties. □

Theorem 4.2.5. For any $t \in N$, there exists $s: V \to N$ which is optimal for t. Moreover, s has the additional properties:

(a) if e_1 and e_2 are two edges with $\text{high}(e_1)$ $= \text{high}(e_2)$, then $|\text{total}(e_1,s) - \text{total}(e_2,s)| \leq 1$,

(b) if e is any edge with $\text{high}(e)$ at level k, then $\text{total}(e,s) \leq \lceil \frac{t}{2^k} \rceil$.

Proof. Theorem 4.2.4 leads naturally to a recursive algorithm yielding a placement in $V \to N$ and obviously satisfying (a). To see that (b) is also satisfied, assume the contrary, and let e be a highest edge in the tree with $\mathrm{total}(e,s) > \left\lceil \frac{t}{2^k} \right\rceil$, where $\mathrm{high}(e)$ is at level k.

Since $\mathrm{total}(e,s) \in N$, we have $\mathrm{total}(e,s) \geq \left\lceil \frac{t}{2^k} \right\rceil + 1 \geq \frac{t}{2^k} + 1$. e is not $\mathrm{rootedge}(T)$, so let e' be the edge $\neq e$ with $\mathrm{high}(e) = \mathrm{high}(e')$, and let e'' be the edge immediately above e in T. Then

$$\mathrm{total}(e'',s) \leq \left\lceil \frac{t}{2^{k-1}} \right\rceil < \frac{t}{2^{k-1}} + 1.$$

Therefore,

$$\mathrm{total}(e's) \leq \mathrm{total}(e'',s) - \mathrm{total}(e,s)$$

$$< \left(\frac{t}{2^{k-1}} + 1\right) - \left(\frac{t}{2^k} + 1\right) = \frac{t}{2^k}.$$

But then

$$|\mathrm{total}(e,s) - \mathrm{total}(e',s)| > 1,$$

contradicting property (a). □

Once again, we analyze the cost of determining an optimal placement with the properties of Theorem 4.2.5. One must calculate $\mathrm{expflow}_0(t)$, and also $\mathrm{expflow}_1(i)$ for all $i \in N$, $i \leq \lceil t/2 \rceil$, $\mathrm{expflow}_2(i)$ for all $i \in N$, $i \leq \left\lceil \frac{\lceil t/2 \rceil}{2} \right\rceil = \lceil t/4 \rceil, \ldots,$ $\mathrm{expflow}_{\log(n)}(i)$ for all $i \in N$, $i \leq \lceil t/n \rceil$. This is a total of $O(t)$ expflow computations, (since we are assuming that n is $O(t)$).

Theorem 4.2.6. There is an algorithm using $O(t^2)$ arithmetic operations, which for any tree T and for any $t \in N$, determines an $s: V \to N$ which is optimal for t, and which satisfies conditions (a) and (b) of Theorem 4.2.5.

4.3. A Fast Algorithm for Determining Optimal Placements

The results of Section 2.4 can be used to prune the algorithm's search space still further, leading to a much faster algorithm.

Theorem 4.3.1. Let t, u, $k \in N$, $k \leq \log(n) - 1$, $\left\lfloor \frac{t}{2^k} \right\rfloor \leq u \leq \left\lceil \frac{t}{2^k} \right\rceil$. Then

$$\mathrm{minexpcost}_k(u) = \mathrm{expflow}_k(u)$$
$$+ \min\{\mathrm{minexpcost}_{k+1}(u_1) + \mathrm{minexpcost}_{k+1}(u_2) \mid u_1, u_2, \in \{\left\lfloor \frac{t}{2^{k+1}} \right\rfloor, \left\lceil \frac{t}{2^{k+1}} \right\rceil\} \text{ and } u_1 + u_2 \leq u\}.$$

Proof. Again, we show $\geq$. Write f for $\mathrm{minexpcost}_{k+1}$. By Theorem 4.2.4, there is a pair u_1, $u_2 \in N$ such that $u_1 + u_2 \leq u$, $|u_2 - u_1| \leq 1$ and $\mathrm{minexpcost}_k(u) = \mathrm{expflow}_k(u) + f(u_1) + f(u_2)$. Of all such pairs, choose one minimizing $u - (u_1 + u_2)$ and assume $u_1 \leq u_2$. We must check that u_1, $u_2 \in \{\left\lfloor \frac{t}{2^{k+1}} \right\rfloor, \left\lceil \frac{t}{2^{k+1}} \right\rceil\}$.

If $u_2 > \left\lceil \frac{t}{2^{k+1}} \right\rceil$, then $u_2 \geq \left\lceil \frac{t}{2^{k+1}} \right\rceil + 1$, so $u_1 \geq \left\lceil \frac{t}{2^{k+1}} \right\rceil$. Then $u \geq u_1 + u_2 \geq 2\left\lceil \frac{t}{2^{k+1}} \right\rceil + 1 > \left\lceil \frac{t}{2^k} \right\rceil$, a contradiction. Thus, u_2 (and therefore u_1) $\leq \left\lceil \frac{t}{2^{k+1}} \right\rceil$.

If $u_1 < \left\lfloor \frac{t}{2^{k+1}} \right\rfloor$, then $u_1 \leq \left\lfloor \frac{t}{2^{k+1}} \right\rfloor - 1$, so $u_2 \leq \left\lfloor \frac{t}{2^{k+1}} \right\rfloor$. Then $u - (u_1 + u_2) \geq \left\lfloor \frac{t}{2^k} \right\rfloor - \left(\left\lfloor \frac{t}{2^{k+1}} \right\rfloor - 1 + \left\lfloor \frac{t}{2^{k+1}} \right\rfloor\right) \geq 1$. Define a new decomposition w_1, w_2 by $w_1 = u_1 + 1$, $w_2 = u_2$. Then $w_1 + w_2 \leq u$ and $|w_2 - w_1| \leq 1$. Now, $\mathrm{expflow}_{k+1}(w_1) \leq \mathrm{expflow}_{k+1}(u_1)$ because $u_1 \leq \left\lfloor \frac{t}{2^{k+1}} \right\rfloor - 1$, by Theorems 2.2.3, 2.4.1 and 2.4.2. Thus, $f(w_1) \leq f(u_1)$, by Theorem 2.2.4. Hence, w_1, w_2 also satisfies the conditions used in choosing u_1, u_2, but $u - (u_1 + u_2) > u - (w_1 + w_2)$, contradicting the minimality condition. □

Theorem 4.3.2. For the special case of this section, there exists a fair whole placement s which is optimal for t.

Proof. Theorem 4.3.1 yields a recursive algorithm. □

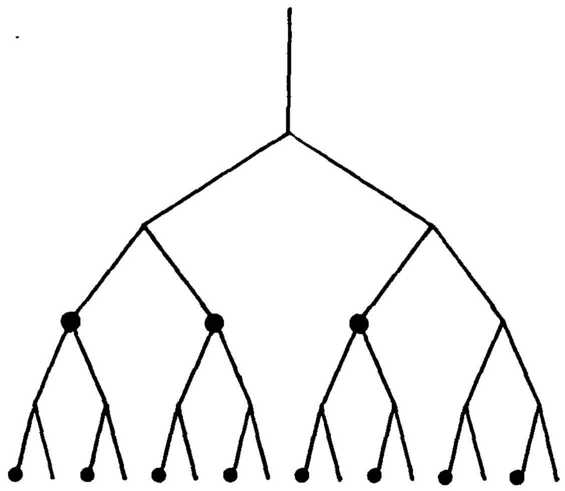
Note that a result similar to Theorem 4.3.2 does not hold in the general case -- recall Example 2.4.1.

The algorithm resulting from Theorem 4.3.2 is extremely fast. Namely, one must calculate $\mathrm{expflow}_k(i)$ for $i = \left\lfloor \frac{t}{2^k} \right\rfloor$, $\left\lceil \frac{t}{2^k} \right\rceil$, for each k, $1 \leq k \leq \log(n)$, for a total of only $O(\log n)$ expflow computations.

Theorem 4.3.3. There is an algorithm using $O(t \log n)$ arithmetic operations which for any tree T and for any $t \in N$, determines a fair whole placement s which is optimal for t.

4.4. Example

Some of the optimal placements discovered by our algorithms are rather unexpected. For example, the following represents an optimal placement of 11 resources in a balanced binary tree with uniform probability distribution:



Acknowledgements

The authors thank Carl Spruill, Charles Blair and Mike Paterson for contributing their ideas and suggestions for some of the results in this paper.

References

1. K. Jogdeo and S.M. Samuels, "Monotone Convergence of Binomial Probabilities and a Generalization of Ramanujan's Equation," The Annals of Mathematical Statistics 39, 4 (1968), 1191-1195.
2. W. Uhlmann, "Ranggrössen als Schätzfunktionen," Metrika 7, (1963), 23-40.
3. W. Uhlmann, "Vergleich der hypergeometrischen mit der Binomial-Verteilung," Metrika 10, (1966), 145-158.

DISTRIBUTION LIST

Office of Naval Research Contract N00014-80-C-0221
Michael J. Fischer, Principal Investigator

Defense Documentation Center
Cameron Station
Alexandria, VA 22314
(1 copy)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Dr. R. B. Grafton, Scientific Officer (1 copy)
Information Systems Program (437) (2 copies)
Code 200 (1 copy)
Code 455 (1 copy)
Code 458 (1 copy)

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106
(1 copy)

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375
(6 copies)

Office of Naval Research
Resident Representative
University of Washington, JD-27
422 University District Building
1107 NE 45th Street
(1 copy)

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380
(1 copy)

Naval Ocean Systems Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152
(1 copy)

Mr. E. H. Gleissner
Naval Ship Research and Development Center
Computation and Mathematics Department
Bethesda, MD 20084
(1 copy)

Captain Grace M. Hooper (008)
Naval Data Automation Command
Washington Navy Yard
Building 166
Washington, D.C. 20374
(1 copy)

Defense Advanced Research Projects Agency
ATTN: Program Management/MIS
1400 Wilson Boulevard
Arlington, VA 22209
(3 copies)

Professor Nancy A. Lynch
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
(1 copy)

Professor Nancy Griffeth
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
(1 copy)

Dr. Leo J. Guibas
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
(1 copy)

Professor Philip Enslow
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
(1 copy)

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, IL 60605
(1 copy)

Dr. John Cherniavsky
Program Director
Theoretical Computer Science
National Science Foundation
Washington, D.C. 20550
(2 copies)

Department of the Army
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709
(1 copy)

DATE
ILMED
3-8